EE2703 : Applied Programming Lab
Assignment 5
Laplace Equation

Sai Shashank GP
EE20B040

March 16, 2022

0.1 Introduction

In this assignment, we have to plot different plots of electric potential in a resistive plate connected to ground and 1V at different points. We are going to use **pylab** for plotting 2D plots. We are going to use **axes3d** class for plotting 3D plots.

0.2 Assignment

0.2.1 Prerequisites

First, we are going to import some useful libraries and classes

```
import sys
from pylab import *
import numpy as np
import mpl_toolkits.mplot3d.axes3d as p3
from scipy.linalg import lstsq
```

0.2.2 Initial Potential Distribution

When we establish the connection, the potential is can be easily predicted. It will be 1V at the contact points of 1V wire and 0V elsewhere.
This can be represented as a matrix of potentials where the matrix is analogous to the resistive plate. So, the following code does that

```
# Taking inputs for Nx, Ny, radius, Niter

Nx = 25 # int(sys.argv[1])       # Default should be 25
Ny = 25 # int(sys.argv[2])       # Default should be 25
radius = 8 # int(sys.argv[3])   # Default should be 8
Niter = 1500 # int(sys.argv[4])    # Default should be 1500

# Initialising the potential array
phi = np.zeros((Nx, Ny))

# Finding out what points lie on the 1V region
x = np.linspace(-1* Nx//2, Nx//2, Nx)
y = np.linspace(-1* Ny//2, Ny//2, Ny)

Y, X = np.meshgrid(y, x)

phi = np.where(X*X+Y*Y <= radius*radius, 1.0, 0.0)
ii = np.where(X*X + Y*Y <= 64)

# plot the contour plot of phi in initial iteration
figure(0)
plot(ii[0]-Nx//2, ii[1]-Ny//2, 'ro')
contour(Y, X, phi)
```

```
xlabel('x')
ylabel('y')
title('Initial Potential Distribution')
show()
```

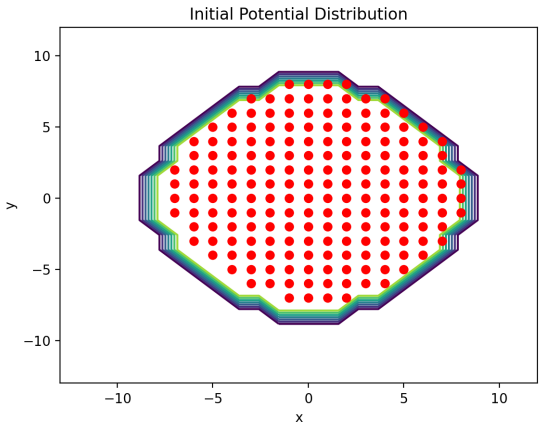

Figure 1: Initial Potential Distribution

0.2.3 Updating Potential Matrix

We use **Laplace Equation** to find out the final distribution of potential. This is achieved from the following equations

$$\nabla^2 \phi = 0 \tag{1}$$

The same in matrix terms is,

$$\phi_{i,j} = \frac{\phi_{i-1,j} + \phi_{i,j-1} + \phi_{i+1,j} + \phi_{i,j+1}}{4} \tag{2}$$

The following code does that for given **Niter** iterations

```
def updatePhi(phi):
    '''Updates phi matrix taking boundary conditions into consideration as well'''
    # For non-boundary points
    phi[1:-1, 1:-1] = 0.25*(phi[1:-1, 0:-2] + phi[0:-2, 1:-1] + phi[1:-1, 2:] + phi[2:,
    # For boundary points
    phi[1:-1, 0] = phi[1:-1, 1]
    phi[1:-1, -1] = phi[1:-1, -2]
    phi[0, 1:-1] = phi[1, 1:-1]
    phi[-1, 1:-1] = 0
    # Correcting the potential at the contact points
    phi[ii] = 1

errors = np.zeros(Niter)

for i in range(Niter):
```

```
    oldphi = np.copy(phi)
    updatePhi(phi)
    errors[i] = np.max(np.abs(oldphi-phi))

# Plot the error trend wrt iterations
figure(1)
semilogy(np.arange(0, Niter, 1)[::50], errors[::50], 'ro')
xlabel('No.of Iterations')
ylabel(r'${error}_i$')
title('Semilog Plot of Error v/s Iterations')
show()

figure(2)
loglog(np.arange(0, Niter, 1)[::50], errors[::50], 'ro')
xlabel('No.of Iterations')
ylabel(r'${error}_i$')
title('Loglog Plot of Error v/s Iterations')
show()
```

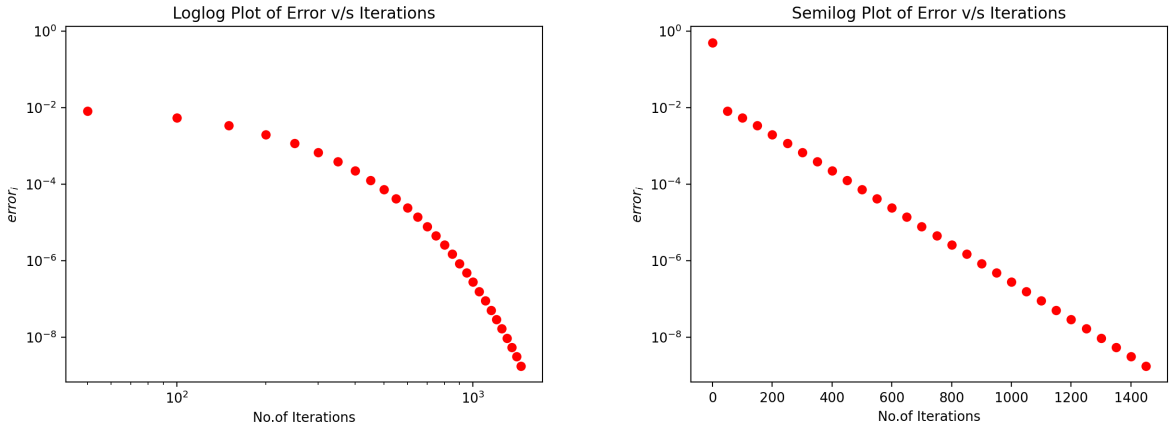

Figure 2: Plots of error v/s iterations

0.2.4 Estimating the error using least squares method

As we can see that the semilog plot of error is linear. Which suggests that error $e(x)$, where x is iteration count, is of the form Ae^{Bx}.
So, in semilog system, It will be $loge = logA + Bx$. We can find A, B using **scipy.linalg.lstsq**. The following code does that

```
def errorFit(x, logA, B):
    return np.exp(logA + B*x)

M1 = np.ones((Niter, 2))
M1[:, 1] = np.arange(0, Niter, 1)
b1 = np.log(errors)
```

```
coeffs_alliter, *f = lstsq(M1, b1)

M2 = np.ones((Niter-500, 2))
M2[:, 1] = np.arange(500, Niter, 1)
b2 = np.log(errors[500:])

coeffs_above500iter, *f = lstsq(M2, b2)

figure(3)
semilogy(np.arange(0, Niter, 1)[::50], errors[::50], 'ro', label='actual')
semilogy(np.arange(0, Niter, 1)[::50], errorFit(np.arange(0, Niter, 1), coeffs_alliter[0
xlabel('No.of Iterations')
ylabel(r'${error}_i$')
legend()
title('Semilog Plot of Fit no. 1 and Actual errors')
show()

figure(4)
semilogy(np.arange(0, Niter, 1)[::50], errors[::50], 'ro', label='actual')
semilogy(np.arange(0, Niter, 1)[::50], errorFit(np.arange(0, Niter, 1), coeffs_above500i
xlabel('No.of Iterations')
ylabel(r'${error}_i$')
legend()
title('Semilog Plot of Fit no. 2 and Actual errors')
show()
```

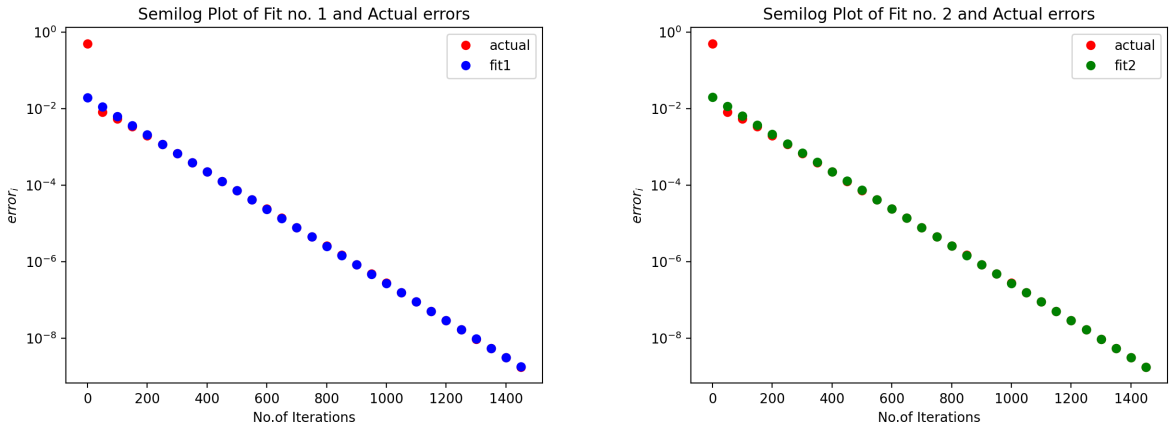

Figure 3: Plots of actual error v/s estimated errors

0.2.5 Stopping point Analysis

In these type of problems, we generally take higher number of iterations to get the most realistic output. But high iterations at all times is not possible. So, we try to know when we can 'stop'

iterating so that it will be a good enough approximation to the final state.
This is mathematically calculated from the **convergence of cumulative error**. In our case the cumulative error upon calculation, has an upper bound as following.

$$Error(k) = \sum_{n=k+1}^{\infty} Ae^{Bn} \tag{3}$$

$$Error(k) = -\frac{A}{B} exp(B(k + 0.5)) \tag{4}$$

As we can clearly see that this doesn't converge to a value. So, we need to perform as much iterations as we can.

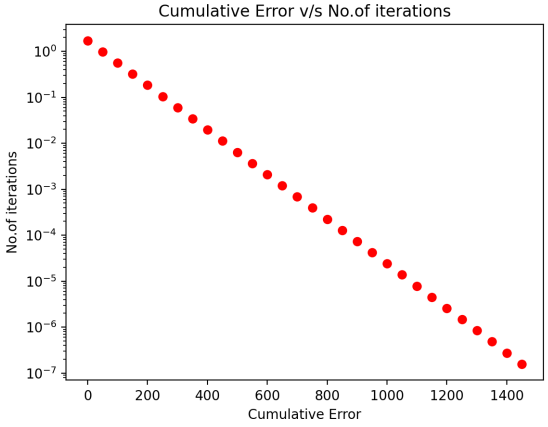

Figure 4: Stopping point Analysis

0.2.6 Plots of the final state

Here are the final state plots that explain the state of resistive plate clearly

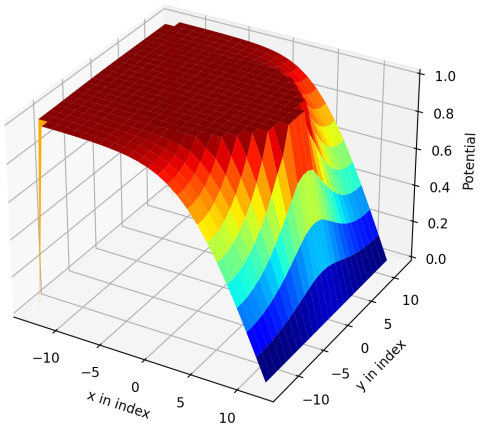

Figure 5: 3-D Surface Plot of potential

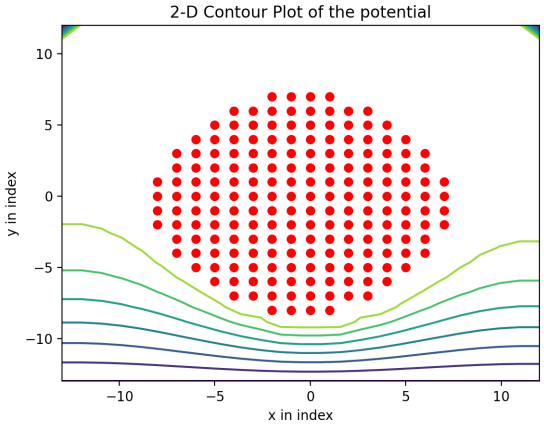

Figure 6: 2-D contour plot of potential

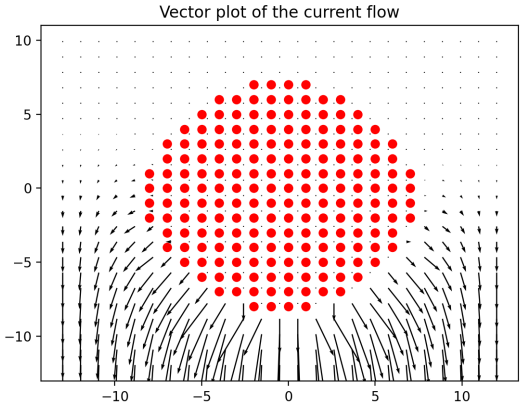

Figure 7: Quiver plot of current densities

0.2.7 Conclusion

We have iterated for a high number of times to get a best estimate of potential at each point on the resistive plate. But as we have seen, this is not the steady state. it is rather an estimate of steady state.
Similarly, to find the temperature at each point on the resistive plate, we can use

$$\nabla(\kappa \nabla T) = \frac{1}{6}|J|^2 \tag{5}$$

But it can be approximated that lower region gets hotter than upper region of the plate since more current flows there.

0.3 TakeAways

From this assignment, I have

- Understood Poisson's equation and Laplace equation clearly and how to implement them in codes

- Understood the effect of boundary conditions very clearly
- Got to know how to do analysis of errors and how important it is
- Got introduced to several plots like surface plot and quiver plot